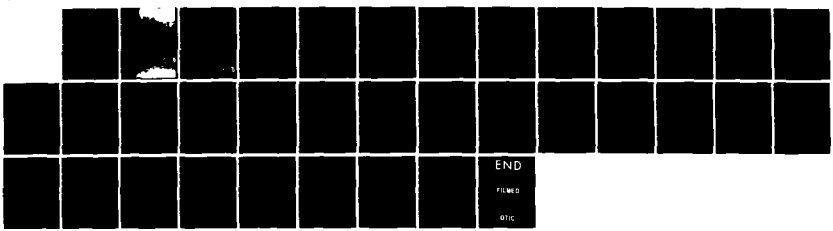
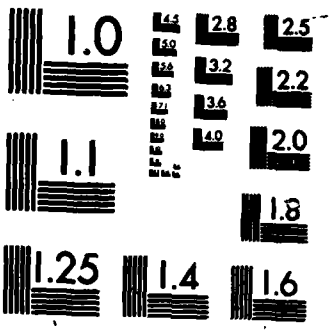
AD-A150 832 A PLAN RECOGNITION MODEL FOR SUBDIALOGUES IN CONVERSATIONS(U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE D J LITMAN ET AL. NOV 84 TR-141 1/1

UNCLASSIFIED N00014-80-C-0197 F/G 5/7 NL





A Plan Recognition Model for Subdialogues in Conversations

Diane J. Litman and James F. Allen
Department of Computer Science
University of Rochester, Rochester, NY 14627

TR 141
November 1984

2

A Plan Recognition Model for Subdialogues in Conversations

Diane J. Litman and James F. Allen
Department of Computer Science
University of Rochester, Rochester, NY 14627

TR 141
November 1984

Abstract

Previous plan-based approaches to analyzing task-oriented dialogues have not been able to account for many phenomena of concern to discourse analysts. We have developed a model based on a hierarchy of plans and meta-plans that accounts for such interrupting subdialogues as clarifications and corrections, while maintaining the advantages of the plan-based approach.

The preparation of this paper was supported in part by the National Science Foundation under Grant IST-8210564, the Office of Naval Research under Grant N00014-80-C-1097, and the Defense Advanced Research Projects Agency under Grant N00014-82-K-0193. A simplified, shortened version appears in the Proceedings of the 10th International Conference on Computational Linguistics, Stanford University, July 1984.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 141	2. GOVT ACCESSION NO. AD-A150 832	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Plan Recognition Model for Subdialogues in Conversations		5. TYPE OF REPORT & PERIOD COVERED technical report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Diane J. Litman and James F. Allen		8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0197 N00014-82-K-0193
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department University of Rochester Rochester, NY 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE November 1984 13. NUMBER OF PAGES 31
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

none

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

natural language, dialogues, discourse structures, problem solving, planning, speech acts, plan recognition, meta-plans, plan stacks, clarification subdialogues,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Previous plan-based approaches to analyzing task-oriented dialogues have not been able to account for many phenomena of concern to discourse analysts. We have developed a model based on a hierarchy of plans and meta-plans that accounts for such interrupting subdialogues as clarifications and corrections, while maintaining the advantages of the plan-based approach. Originator-supplied Keywords:

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1. Introduction

Task-oriented dialogues occur when two people work cooperatively on a task (e.g., a plan) which is performed during the dialogue. One promising approach to analyzing such dialogues has involved modeling the plans of the speakers in the task domain. In other words, the participants in the conversation are viewed as accomplishing goals via plans containing the utterances of the conversation as actions in the plan. The earliest work in this area involved tracking the topic of a dialogue by tracking the progress of the plans of the conversants [Grosz, 1977], as well as explicitly incorporating speech acts into a planning framework for single utterances [Cohen and Perrault, 1979; Allen and Perrault, 1980]. A good example of the current status of these approaches in providing dialogue coherence can be found in [Carberry, 1983]. In general, these models work well as long as the topic follows the task structure closely, but encounter difficulty accounting for subdialogues, such as clarifications and corrections, and topic change.

Sidner and Israel [1981] suggest a solution to a class of subdialogues that correspond to debugging the plan in the task domain. They allow utterances to talk about the task plan, rather than always being a step in the plan. Using their suggestions, as well as our early work [Allen et al., 1982; Litman, 1983], we have developed a model based on a hierarchy of plans and meta-plans that accounts for the debugging subdialogues they discussed, as well as other forms of clarification.

Reichman [1981] suggested a model of discourse that addresses subdialogues and topic switch in unconstrained spontaneous discourse. Unfortunately, she does not show how one can recognize the structures in her abstract model from the actual utterances. Although not the focus of this paper, we claim that a plan recognition model provides the link from the processing of actual input to its abstract discourse structure. Even more important, this allows us to augment our plan recognition model with the linguistic results from such work.

Consider the following two dialogue fragments. Dialogue 1 was collected at an information booth in a train station in Toronto [Horrigan, 1977] and is similar to "information-seeking" dialogues [Carberry, 1983], dialogues in which an agent seeks information with respect to a plan which will not be executed during the dialogue. Dialogue 2 is a scenario developed from protocols in a graphics system that displays network structures [Sidner and Bates, 1983]. Unlike Dialogue 1, the system's interaction with the user is primarily non-linguistic, with utterances only being produced to satisfy simple conversational conventions.

1) Passenger: The eight-fifty to Montreal?
2) Clerk: Eight-fifty to Montreal. Gate seven.
3) Passenger: Where is it?
4) Clerk: Down this way to the left. Second one on the left.
5) Passenger: OK. Thank you.

Dialogue 1

6) User: Show me the generic concept called "employee."
7) System: OK. <system displays network>
8) User: I can't fit a new IC below it. Can you move it up?
9) System: Yes. <system displays network>
10) User: OK, now make an individual employee concept whose first name is "Sam" and whose last name is "Jones." The Social Security number is 234-56-7899.
11) System: OK.

Dialogue 2

While still task-oriented, these dialogues illustrate phenomena characteristic of spontaneous conversation. Subdialogues correspond not only to subtasks (utterances (6)-(7) and (10)-(11)), but also to clarifications ((3)-(4)), debugging ((8)-(9)), and other types of topic switch and resumption. Furthermore, since these are extended discourses rather than unrelated question/answer exchanges, participants need to use information provided by previous utterances. For example, (3) would be difficult to understand without the discourse context of (1) and (2). Finally, these dialogues illustrate the following of conversational conventions such as terminating dialogues with typical closing expressions (utterance (5)) and the use of words like "OK."

To address these issues, we present a plan-based natural language system that incorporates both task and discourse knowledge. In particular, we develop a new model of plan recognition that accounts for the recursive nature of plan suspensions and resumptions. Section 2 presents this model, followed in Section 3 by a brief description of the discourse analysis performed and the task and discourse interactions. Sections 4 and 5 trace the processing of Dialogues 1 and 2 in detail, illustrating domain independence. This work is then compared to previous work in Section 6, and summarized in Section 7.

2. Task Analysis

2.1 The Plan Structures

In addition to the standard domain-dependent knowledge of task plans, we introduce some knowledge about the planning process itself. These are domain-independent plans that refer to the state of other plans. Plans involve both physical and linguistic actions, each with the system and the user as possible agents. In a typical plan involving a conversation, for example, each agent takes turns executing actions corresponding to the utterances in the conversation. During a dialogue, we shall recognize such plans and put them on a stack, each plan on the stack referring to the plan below it, with the domain-dependent task plan at the bottom. As we shall see, the manipulation of this stack of plans is similar to the manipulation of topic hierarchies that arise in discourse models.

To allow plans about plans, i.e., *meta-plans*, we need a vocabulary for referring to and describing plans. Developing a fully adequate formal model would be a large research effort in its own right. Our development so far is meant to be suggestive of what is needed, and is specific enough for our preliminary implementation. Also, for the purpose of this paper, we are ignoring all temporal qualifications (e.g., the constraints need to be temporally qualified), and all issues involving beliefs of agents.

Plans are networks of actions and states connected by causality and subpart relationships. Every plan has a *header*, a parameterized action description that names the plan. The *parameters of a plan* are the parameters in the header. As usual in many models of planning (e.g., [Fikes and Nilsson, 1971; Sacerdoti, 1977]), plans may contain *prerequisites*, *effects*, and a *decomposition*. In other words, plans are hierarchical. Although a plan may be usefully thought of at one level of abstraction as a single action, the plan may in actuality be composed of primitive actions and other abstract action descriptions (i.e., other plans). Thus, decompositions may be sequences of actions, sequences of subgoals to be achieved, or a mixture of both. We will ignore most prerequisites and effects thoughout this paper, except when needed in examples.

Associated with each plan is a set of *constraints*. These are similar to prerequisites, except that the planner never attempts to achieve a constraint if it is false. Thus, any action whose constraints are not satisfied in some context will not be applicable in that context.

For example, the first plan in Figure 1 summarizes a simple plan schema with a header "BOARD (agent, train)," with parameters "agent" and "train," and with the constraint "EQUAL (depart-station (train), Toronto)." This constraint captures the knowledge that the information booth, and hence the agent, is in the Toronto station. Again, note that constraints are not treated as goals to be achieved. Thus this plan schema would best be described in English as "boarding a train in Toronto" rather than as the general action of "boarding a train." The plan's decomposition consists of the three steps indicated. The prerequisites and effects are not shown. The second plan indicates a primitive action, GOTO, and its effect. Other plans needed in this domain include plans to meet trains, plans to buy tickets, etc.

```
---------------------------------------------------------------------------
HEADER: BOARD (agent, train)
DECOMPOSITION: BUY-TICKET (agent, train)
               GOTO (agent, depart-location (train),
                           depart-time (train))
               GETON (agent, train)
CONSTRAINTS: EQUAL (depart-station (train), Toronto)
---------------------------------------------------------------------------
HEADER: GOTO (agent, location, time)
EFFECT: AT (agent, location, time)
---------------------------------------------------------------------------
HEADER: MEET (agent, train)
DECOMPOSITION: GOTO (agent, arrive-location (train),
                           arrive-time (train))
CONSTRAINTS: EQUAL (arrive-station (train), Toronto)
---------------------------------------------------------------------------
```

Figure 1: Domain Plan Schemas

Throughout this paper we assume that formulas are expressed in a typed logic that will be reflected in the naming of variables. Thus, in the above formulas, *agent* is restricted to entities capable of agency, *train* is restricted to trains, *plan* is restricted to objects that are plans, etc. As we shall see later, types are organized into type hierarchies as commonly found in semantic network formalisms.

Plan schemas can be used for both plan generation and plan recognition. For example, a planner would use these schemas in the same way it would have used STRIPS action descriptions [Fikes and Nilsson, 1971], e.g., to generate sequences of instantiated schemas to achieve some goal. Once generated, the complex plan is executed much as one would run a program. A plan recognizer, on the other hand, will use the plan schemas to recognize the plan instantiation which produced an executed action. In particular, the recognizer will be concerned with recognizing plans from actions executed as part of a dialogue.

Plans about plans, or meta-plans, deal with introducing plans, executing plans, specifying parts of plans, debugging plans, abandoning plans, etc., independently of any domain. To talk about the structure of plans we will assume the predicate *PARAMETER (P, plan)*, which asserts that P is a parameter of the specified plan. We also use a predicate *STEP* (*action, plan*), which asserts that the specified action is a step of the specified plan. Other predicates will be introduced as they are needed.

Plans are not the only objects whose structure we need to examine. In addition, we will need to refer to parameters of actions and propositions as well. Thus, we will be working in a logic admitting plans, actions, and propositions as objects. The PARAMETER predicate will be used to make assertions about the structure of all these types of objects.

Other than the fact that they refer to other plans, meta-plans are identical in structure to domain plans. Two examples of meta-plans are given in Figure 2. The first one, INTRODUCE-PLAN, takes a plan of the speaker which involves the hearer and presents it to the hearer, who is assumed to be cooperative. One way of doing this is to request one of the actions in the plan for which the hearer is the agent. The definitions of the speech acts will be provided in the next section. The second meta-plan, IDENTIFY-PARAMETER, provides a suitable description of the *parameter* that enables the *hearer* to execute the *action* as part of the *plan*. It has several constraints on the relationship between the meta-plan and the plan it concerns, namely that *parameter* must be a parameter of an *action* which must be in the *plan*, and that the description will involve the specification of *term*. We define the predicate KNOW-PARAMETER (agent, term, action, plan) to mean that the *agent* has a description of the *term* that is informative enough to allow the agent to execute the *action* in the *plan*, all other things being equal. While the axiomatization of KNOW-PARAMETER is problematic, we shall only be using it in simple cases where its use is straightforward. Finally, the predicate NEXT interacts with the discourse analysis and will be explained in a later section.

There are many more meta-plans concerning plan debugging, plan execution, etc., some of which will be discussed in the example in Section 5.

During a dialogue, a stack of executing and suspended plans is built, each plan referring to the plan below it, with the domain-dependent task plan at the bottom and the currently executing plan at the top. Other models of discourse (e.g., [Reichman, 1981; Polanyi and Scha, 1983]), have shown that the topic structure follows a stack-like discipline.

```
HEADER: INTRODUCE-PLAN (speaker, hearer, action, plan)
DECOMPOSITION: REQUEST (speaker, hearer, action)
EFFECTS: WANT (hearer, plan)
         NEXT (action, plan)
CONSTRAINTS: STEP (action, plan)
             AGENT (action, hearer)
```

```
HEADER: IDENTIFY-PARAMETER (speaker, hearer,
            parameter, action, plan)
DECOMPOSITION: INFORMREF (speaker, hearer, term, proposition)
EFFECTS: NEXT (action, plan)
         KNOW-PARAMETER (hearer, parameter, action, plan)
CONSTRAINTS: PARAMETER (parameter, action)
             STEP (action, plan)
             WANT (hearer, plan)
             PARAMETER (parameter, proposition)
             PARAMETER (term, proposition)
```

Figure 2: Meta-plan Schemas

In this paper, the stack of plans will always represent what the system believes is the state of the joint plan. Because both agents may construct and execute these plans, however, at times it will seem that the stack is not truly a stack. This occurs when the user acts and the system has to recognize what sequence of planning and execution steps the user did. For example, if the user popped the top plan, and executed a step in what is now the user's top plan, the system would recognize this as executing a plan in the second from the top plan. This anomaly is quickly resolved as the system can then pop its stack to bring the two agents' views back into synchronization. Thus, once the plan recognition process is completed, the observed action is always in the plan that is on the top of the stack.

To rephrase this, plans are added and deleted according to the stack discipline. The plan recognizer, however, is allowed to inspect the entire stack in order to recognize that the user has popped the stack before the user executed the recognized action. Even when the system believes the top plan has completed successfully, it cannot be popped before some acknowledgement from the user, thus allowing for a clarification of the complete plan. The acknowledgement could be explicit, but most often is implicit in that the user acts in such a way that the system recognizes that it must pop the top plan.

The state of the stack can thus be viewed as having the general structure shown in Figure 3. At the top of the stack there is a set of plans (possibly null) which the recognizer believes has been executed and completed. Below these plans are the currently suspended plans. Each suspended plan will be resumed when the one above it is popped. In the case when there are no completed plans, the top of the suspended plans is also believed to be executing. Thus, the top of the stack is either an executing or just-executed plan. The actual pushing and popping of plans will be discussed in Section 2.3 (plan recognition).

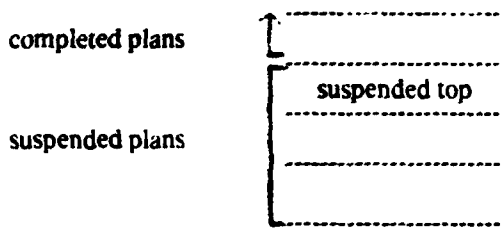


Figure 3: The General Form of the Plan Stack

As an example, a clarification subdialogue is modeled by a plan structure that refers to the plan that is the topic of the clarification. When a clarification plan is recognized, it is pushed onto the stack. The previous top, the plan being clarified, is temporarily suspended. The stack is shown in Figure 4. When the clarification is complete and its success acknowledged, the stack is popped and resumption of the previous plan is recognized.

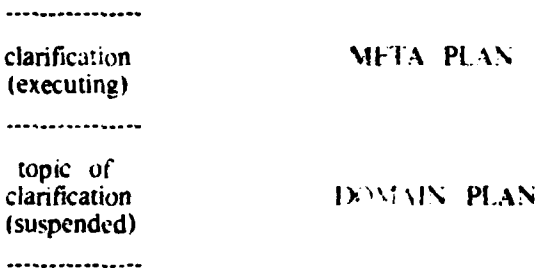


Figure 4: A Clarification Plan Stack

2.2 Speech Act Definitions

We are assuming suitable definitions of the speech acts as in Allen and Perrault [1980], shown in Figure 5. We have modified decompositions to include the conventionalized forms of indirect speech acts. For example, the second decomposition of the REQUEST for an action A is that the speaker REQUEST the hearer to INFORMIF the hearer can do A. Such an inference could be derived from first principles [Allen and Perrault, 1980]. We will not address those issues here.

We have also explicitly added an effect that the hearer then believes the preconditions held if the act is done successfully. This could again be inferred from first principles, but adding it to the definition allows us to use a simple plan recognition algorithm throughout the paper.

HEADER: REQUEST (speaker, hearer, action)
EFFECT: WANT (hearer, action)
KNOW (hearer, WANT (speaker, action))
CONSTRAINTS: AGENT (action, hearer)
DECOMPOSITION-1: S-REQUEST (speaker, hearer, action)
DECOMPOSITION-2: S-REQUEST (speaker, hearer, INFORMIF (hearer, speaker, CANDO (hearer, action)))
DECOMPOSITION-3: S-INFORM (speaker, hearer, ~(CANDO (speaker, action)))
DECOMPOSITION-4: S-INFORM (speaker, hearer, WANT (speaker, action))

HEADER: INFORM (speaker, hearer, proposition)
PREREQUISITES: KNOW (speaker, proposition)
EFFECT: KNOW (hearer, proposition)
KNOW (hearer, KNOW (speaker, proposition))
DECOMPOSITION: S-INFORM (speaker, hearer, proposition)

HEADER: INFORMREF (speaker, hearer, term, proposition)
PREREQUISITES: KNOWREF (speaker, term, proposition)
DECOMPOSITION: achieve KNOW (hearer, proposition)
EFFECT: KNOWREF (hearer, term, proposition)
CONSTRAINT: PARAMETER (term, proposition)

HEADER: INFORMIF (speaker, hearer, proposition)
PREREQUISITES: KNOWIF (speaker, proposition)
DECOMPOSITION-1: achieve KNOW (hearer, proposition)
DECOMPOSITION-2: achieve KNOW (hearer, ~proposition)
EFFECT: KNOWIF (hearer, proposition)

Figure 5: Speech Act Definitions

The only other difference from Allen and Perrault [1980] is that we have added an extra parameter to the INFORMREF action and the KNOWREF operator. The assertion KNOWREF (A, t, p) means that A knows a description of term t, which satisfies proposition p.

This is simply a notational variant that is closer to the actual implementation. Thus, rather than stating the goal to know when train TR1 leaves as

"KNOWREF (A, the x: depart-time (TR1, x))"

as in Allen and Perrault [1980], we write

KNOWREF (A, ?time, EQUAL (depart-time (TR1), ?time)).

Not all such assertions involve the equality predicate. For example, the representation of the goal behind the utterance "What do you want?" would be

KNOWREF (You, ?action, WANT (You, ?action)).

We can define this operator formally within a possible worlds semantics of the BELIEF operator by using "quantifying in" as done in Allen and Perrault [1980]. While this analysis is not fully satisfactory, it is adequate for our present purposes.

For this paper we are assuming all meta-plans are done using speech acts. For example, another way to achieve KNOWREF goals would have been to look up the answer in a reference source. At the train station, for example, one can find departure times and locations from a schedule.

2.3 Plan Recognition

The plan recognizer attempts to recognize the meta-plan, and thus the related domain or meta-plan that led to the production of the input utterance. An utterance either continues an existing plan on the stack or introduces a meta-plan to some plan on the stack. If either of these is not possible for some reason, the recognizer hypothesizes a plausible plan using any of the plan schemas. At the beginning of a dialogue, the general expectations from the task domain are used to guide the plan recognizer, e.g., a train clerk expects questions about boarding and meeting a train.

The plan recognizer's task is to find a sequence of instantiations of plan schemas, each one containing the previous one in its decomposition, that connects the utterance speech act to an expected goal. More specifically, the system tries to find plans in which the utterance is a step, and then tries to find more abstract plans for which the postulated plan is a step, and so on. If during this process the observed action can be incorporated into a plan according to one of the following three ordered preferences, the chaining stops:

1) by a continuation of the top suspended plan on the stack;
2) by introducing a clarification meta-plan to any plan on the stack;
3) by constructing a plan or plans that are plausible given the domain-specific expectations about plausible goals of the speaker.

The preferences introduce heuristics for choosing the sequence that corresponds to the most coherent continuation of the dialogue. Thus we prefer an utterance interpretation that continues a plan rather than clarifying one, which is more coherent than introducing a new plan altogether. While these heuristics have not been empirically validated, they have intuitive appeal. Other models of discourse (e.g., [Carberry, 1983; McKeown, 1982]) also use similar heuristics.

Candidate plans are also eliminated by a set of heuristics based on those in Allen and Perrault [1980]. For example, plans that are postulated whose effects are already true are eliminated, as are plans whose constraints cannot be satisfied, and plan parameters which are compatible with known objects are asserted to be equal to such objects. Unfortunately, we do not have room to discuss our investigation of the technical issues generated by performing this type of non-monotonic equality reasoning. When heuristics cannot eliminate all but one postulated plan, the chaining stops, even when a solution path has not yet been found; in other words, search terminates after branch points. Such premature termination is typical in dialogue processing (e.g., see [Sidner and Israel, 1981; Carberry, 1983]), since later utterances in the dialogue often eliminate many of the branches. For example, since both BOARD and MEET plans can be recognized from a GOTO (recall Figure 1), chaining would halt with two branches if both plans were plausible. Thus, the bottom-up inference process halts when we have either incorporated the utterance into our expectation structure (satisfied a preference) or the search space becomes

too large. In the latter case, multiple stacks are created to record each branch and the plan remains ambiguous.

Preference (1) above involves situations where the agent does exactly what was expected in the given situation. The most common example of this occurs in answering a question, where the answer is explicitly expected. As another example, if the agent was observed going to the ticket window and paying for a ticket, the BUY plan would be postulated. If the agent is next observed receiving the ticket, it would be recognized as a continuation of that BUY plan. A more complex example is illustrated in Figure 6, where a BUY subplan connects an observed GOTO action with an expected (stacked) BOARD goal.

Figure 6: Introduce a Plausible Subplan to Continue the Executing Plan on the Stack

Preference (2) involves not only recognizing a clarification meta-plan based on the utterance, but also, in satisfying its constraints, connecting the meta-plan to a plan on the stack. If the plan on the stack is not the top plan the stack must be popped down to this plan before the new meta-plan is added to the stack. If the plan that is the object of the clarification is ambiguous, the alternative closest to the top of the stack is preferred. For example, if the BOARD plan expanded from Figure 6 is on the top of the stack, the utterance "How much does the ticket cost?" could be recognized as a request for a clarification of the PAY subplan. The clarification plan would be placed on the stack and the BOARD plan suspended (shown in Figure 7).

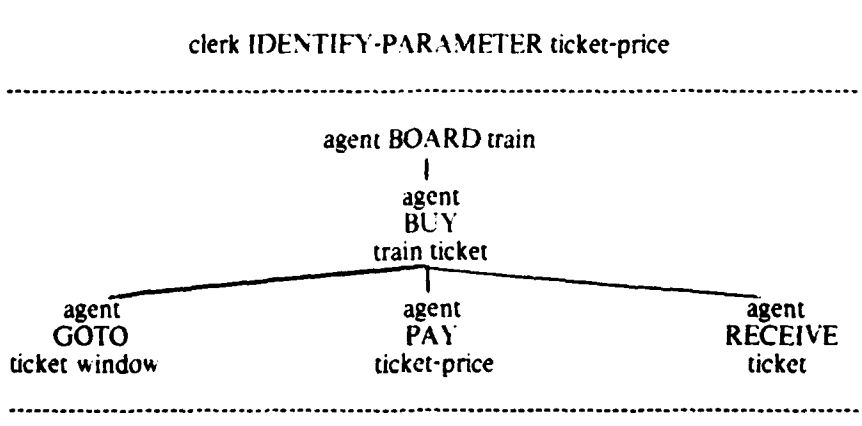


Figure 7: Introduce a Clarification Meta-plan to a Plan on the Stack

Preference (3) may involve not only introducing a plan, but also, if it is a meta-plan, using the constraints to recursively introduce a plausible plan for the meta-plan to be about. This occurs most frequently at the start of a dialogue. Suppose a speaker begins a dialogue with "I want to buy a ticket to Montreal." The utterance is recognized as an explicit plan introduction, a meta-plan with constraints that enable the recognition of the plan being introduced as well. Figure 8 shows the stack constructed out of these plans. As desired, the plans are placed on the stack in the order they were generated rather than the order they were recognized.

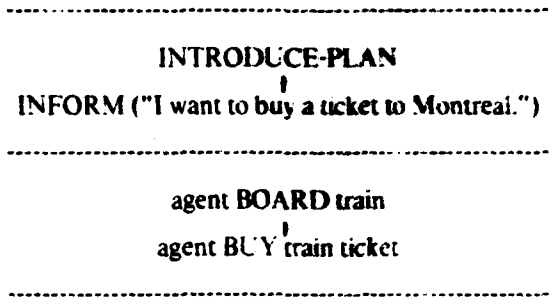


Figure 8: Recognizing Multiple Plans from One Utterance

Note that each preference involves not only recognizing a meta-plan based on the utterance, but in satisfying its constraints, also involves connecting the meta-plan to an expected plan (which is either an already stacked plan or an introduced plausible domain plan).

Once a plan (or set of plans) is recognized it is expanded top-down by adding the definitions of all steps and substeps until there is no unique expansion for any of the remaining substeps.

If there are multiple interpretations remaining at the end of this process, multiple versions of the stack are created to record each possibility. There are then several ways in which one might be chosen over the others. For example, if it is the hearer's turn in the dialogue (i.e., no additional utterance is expected from the speaker), then the hearer may initiate a clarification subdialogue. If it is still the speaker's turn, the hearer may wait for further dialogue to distinguish between the possibilities.

3. Communicative Analysis and Interaction with Task Analysis

Much research in recent years has studied largely domain-independent linguistic issues. Since our work concentrates on incorporating the results of such work into a plan recognition framework, rather than on a new investigation of these issues, we will first present the relevant results and then explain our work in those terms.

Grosz [1977] noted that in task-oriented dialogues the task structure could be used to guide the discourse structure. She developed the notion of *global focus* of attention to represent the influence of the discourse structure; this proved useful for limiting the search space during the resolution of definite noun phrases. Carberry [1983] provided explicit plan recognition rules for tracking shifts in the task structure. From an utterance, she recognized part of the task plan, which was then used to provide expectations for future plan recognition. For example, upon completion of a subtask, execution of the next subtask was the most salient expectation. Global focus needs to be distinguished from *immediate focus* [Grosz, 1977; Sidner, 1983], which represents the influence of the linguistic form of the utterance and proves useful for understanding ellipsis, definite noun phrases, pronominalization, and "this" and "that."

Reichman [1981] developed a model which was not limited to task-oriented dialogues, and accounted for a much wider range of discourse popping (e.g., topic switch). In her theory, she noted that the non-linear structure underlying a dialogue was reflected by the use of surface phenomena such as referring expressions and clue words. The type of expression (e.g., pronoun, definite noun phrase, etc.) used to refer to an object was shown to reflect the object's degree of focus. Clue words signaled a boundary shift between the discourse units hierarchically structured as well as the kind of shift. e.g., the clue word "now" indicated the start of a new unit which further developed the currently active unit.

Our communicative analysis is a step toward incorporating these results on focusing, with some modification, into the plan recognition system. In our framework, each plan on the stack has a marked subplan which is under execution or will be executing when the plan is resumed. This subplan is indicated using predicates as follows:

> NEXT (subplan, plan) -- true only if the *subplan* is the part of the *plan* that is currently being executed, or will be executed when the plan is resumed;
>
> LAST (subplan, plan) -- true if the *subplan* is the last (i.e., most recent) part of the *plan* to be executed.

Among the effects of every meta-plan are assertions that update these predicates with respect to the plan referred to. We will often refer to the subplan that is next as the part of the plan that is *in focus*, drawing an analogy with the discourse models above.

Now we can see that the ordering of the preferences in plan recognition encodes expectations as to the most coherent focus shifts. In particular, the most coherent continuation simply follows the task structure in the top plan, corresponding to preference (1). Following that, a clarification of some completed subplan is expected, corresponding to preference (2).

As in Grosz [1977] and Reichman [1981], we also use surface linguistic phenomena to help determine focus shifts. (See [Polanyi and Scha, 1983; Litman, 1983] for other surface phenomena that mark discourse functions.) For example, clue words often explicitly mark what would be an otherwise incoherent or unexpected focus switch, as when "by the way" is used before starting a temporary topic change. Robin Cohen [1983] uses clue words in a similar way for understanding arguments. Her arguments are analyzed as trees (with nodes connected by the evidence relation, as opposed to our subtask relation); a coherent argument corresponds to specific tree traversal orders unless these default traversals are explicitly overruled by a clue word.

Our meta-plans and stack mechanism capture Reichman's manipulation of the context space hierarchies for topic suspension and resumption. In particular, the stack manipulations can be viewed as corresponding to the following discourse situations. If the plan recognized is already on the stack, then the speaker is continuing the current topic, or is resuming a previous (stacked) topic. If the plan is a clarification meta-plan to a stacked plan, then the speaker is commenting on the current topic, or on a previous topic that is implicitly resumed. In other cases, the speaker is introducing a new topic.

Conceptually, the communicative and task analysis cannot be performed in a fixed sequence. For example, when the task structure is used to guide the discourse structure [Grosz, 1977], plan recognition (production of the task structure) must be performed first. Similarly, such a strategy will be needed in cases where the discourse expectations are violated, such as when questions are ignored. However, suppose the user suddenly changes task plans. Communicative analysis could pick up any clue words signalling this unexpected topic shift, indicating the expectation changes to the plan recognizer. What is important is that either strategy should be dynamically chosen depending on the utterance, in contrast to any a priori sequential (or even cascaded [Brachman et al., 1979]) ordering. The examples below illustrates the necessity of such a model of interaction.

4. Example: Dialogue 1

This section illustrates the system's task and communicative processing of Dialogue 1. As above, we will concentrate on the task analysis; some discourse analysis will be briefly presented to give a feel for the complete system. We will take the role of the clerk, and concentrate on understanding the passenger's utterances.

Our system performs the plan recognition outlined here and is driven by the output of a parser using a semantic grammar [Brown and Burton, 1977] for the train domain. The HORNE Reasoning System [Allen et al., 1983], a lisp-embedded typed horn clause theorem prover, is used as the starting point for our knowledge retrieval mechanism. The incorporation of the discourse mechanism is under development. The system at present does not plan or generate natural language responses.

In order to understand the example, it will be necessary to briefly discuss some aspects of the HORNE system. In particular, HORNE types can have a set of distinguished function names called roles, each with an associated type. For example, the type TrainType is a subtype of the type PhysicalObjectType and has six distinguished (type-restricted) roles, as shown:

```
(subtype TrainType PhysicalObjectType
    (depart-location LocationType)
    (depart-station CityType)
    (depart-time TimeType)
    (arrive-location LocationType)
    (arrive-station CityType)
    (arrive-time TimeType))
```

In other words, role function names are just another type of object to HORNE. We will use the notation ?fn (train1) to represent a role value of train1 (where train1 is an object of type TrainType). Thus, ?fn could be any of the six role names listed above (e.g., depart-location). If we later were to match ?fn(train) with a variable of type LocationType, ?fn(train1) would be further restricted to the depart-location or arrive-location of train1.

Objects of type *proposition* and *action* are also represented in the type hierarchy and have roles defined for each argument, as is done in semantic network representations. Thus, in the implementation, PARAMETER (t, p) is defined to be true only if the term t fills a role of the proposition/action instance p.

Our current system uses a highly-specialized semantic grammar [Brown and Burton, 1977] to parse the utterances in the train domain. This allows us to avoid some difficult parsing issues and concentrate on the plan recognition model. The following analysis of "The eight-fifty to Montreal?" is output from the parser:

```
S-REQUEST (Person1, Clerk1,                                    (R1)
                INFORMREF (Clerk1, Person1, ?term,
                                equal (?term, ?fn (train1))))

with constraints: arrive-station (train1) = Montreal
                  depart-time (train1) = eight-fifty
```

In other words, Person1 is querying the clerk about some (as yet unspecified) term, ?term, that is the value of some role of train1. The parser identifies train1 as a train from the input since trains are the only objects in the domain that are described using times and cities. If there were other possibilities, the parser would need to construct other interpretations as well. S-REQUEST is a surface-request, as defined in Figure 5. In this example, since the utterance is literal rather than indirect, it will later be recognized as a REQUEST.

Since the stack is empty, the plan recognizer can only construct an analysis corresponding to preference (3), where an entire plan stack is constructed based on the domain-specific expectations that the speaker will try to BOARD or MEET a train. From the S-REQUEST, via REQUEST, it recognizes an instantiation of the INTRODUCE-PLAN meta-plan, and from constraint satisfaction we know that the INFORMREF must be a step in the recognized plan. Since INTRODUCE-PLAN is not a step in any plan, chaining stops. Since INTRODUCE-PLAN is a meta-plan, however, we recursively expand the plan containing the INFORMREF, and recognize that it is part of an IDENTIFY-PARAMETER plan.

In satisfying the constraints on IDENTIFY-PARAMETER, a third plan is introduced that must have a step which contains a property of a train, described via the INFORMREF, as its parameter. An eligible domain plan is GOTO, where ?fn(train1) is restricted to be the time or location parameter of the GOTO. As a result of this, ?fn is restricted to be a time or location role of train1. Note that all three plans are recognized before any is placed on the stack. Furthermore, although GOTO is recognized after IDENTIFY-PARAMETER, it is put on the stack first since it is suspended for the IDENTIFY-PARAMETER clarification. The state of the stack after the plan recognition process just discussed is shown in Figure 9, which should be viewed as one stack with three elements: PLAN3 is at the top of the stack, PLAN2 in the middle, and PLAN1 at the bottom. We shall refer to the GOTO action using the name "GO1," and the INFORMREF action using the name "I1."

PLAN3

INTRODUCE-PLAN (Person1, Clerk1, I1, PLAN2)
|
REQUEST (Person1, Clerk1, I1)

PLAN2

IDENTIFY-PARAMETER (Clerk1, Person1, ?fn(train1), GO1, PLAN1)
|
I1: INFORMREF (Clerk1, Person1, ?term, equal (?term, ?fn (train1)))

where ?fn is restricted to be a location or time role of a train, and the type of ?term is restricted to be location or time.

PLAN1

GO1: GOTO (?agent, ?location, ?time)

Figure 9: Constraint Satisfaction Creates PLAN1 and PLAN2

Recursively calling the plan recognizer on PLAN1 selects the BOARD plan; the MEET plan is eliminated due to constraint violation, since the arrive-station is not Toronto. Recognition of the BOARD plan also constrains ?fn to be depart-time or depart-location. Since the effect of the IDENTIFY-PARAMETER plan needs to be achieved, the speaker does not know enough about the ?fn property of the train to execute GO1. Since the depart-time was known from the utterance, the role name depart-time can be eliminated as a possibility for the value of ?fn due to the heuristic that one does not need to execute a plan if the effect is already true. In other words, the plan recognition heuristics discussed above eliminate competing interpretations and constrain ?fn to be the depart-location. Also, since the expected agent of the BOARD plan is the speaker, ?agent is set equal to Person1.

Once the recursive call is completed, plan recognition ends and all postulated plans are expanded top down to include the rest of their steps. The state of the stack is now as shown in Figure 10. As desired, we have constructed an entire plan stack based on the original domain-specific expectations to BOARD or MEET a train.

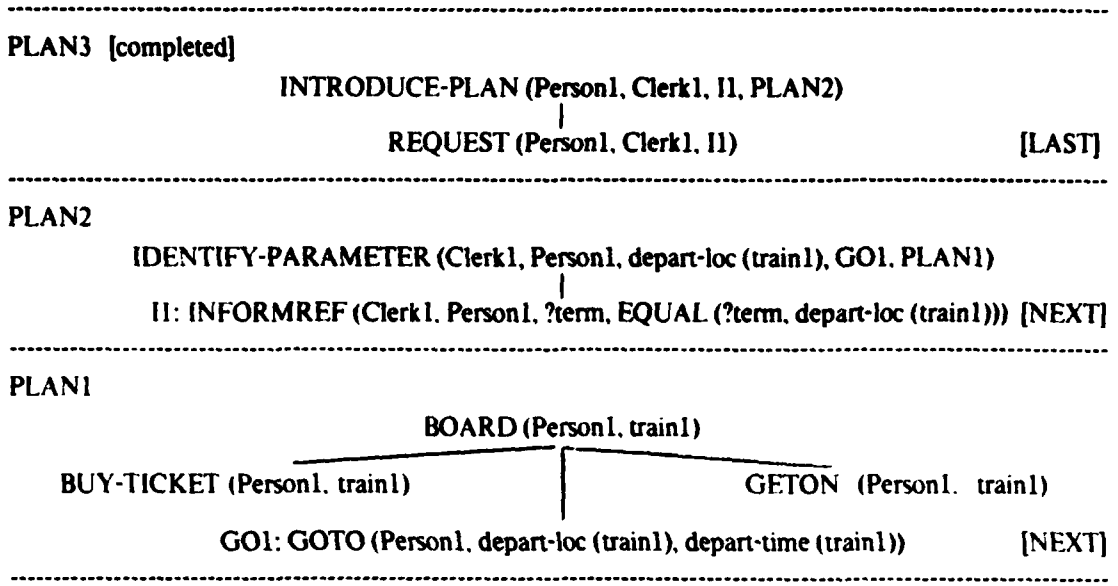


Figure 10: The Plan Stack after the First Utterance

Once the task structure is recognized the global focus (the executing step) in each plan structure is noted, as shown in square brackets in the figures. Thus the REQUEST in PLAN3 is marked as the LAST step of PLAN3. From the effects of INTRODUCE-PLAN and IDENTIFY-PARAMETER we mark I1 and GO1, respectively, as in focus whenever execution returns to those stacked plans.

The clerk's planning of the response is hand simulated in our system. Here we will describe what the system should do. The system responds with the expected continuation, namely the INFORMREF in PLAN2, popping PLAN3 off the stack. This anticipates the passenger's correct interpretation of the utterance, since our stack is assumed to be shared between the speakers. The assumption will remain necessary in this work until we introduce a separate stack for each speaker. To execute the INFORMREF, the system would plan an S-INFORM, say, which might eventually be realized as the utterance "Eight-fifty to Montreal. Gate seven." Using the assumption that this utterance is correctly recognized by the passenger, we update the focus in PLAN2, marking the INFORMREF step as [LAST]. Although all the steps of PLAN2 are completed, its success cannot be confirmed until the next utterance. Thus it is not popped off the stack. This allows the possibility of a clarification plan being introduced concerning PLAN2. Thus, at the stage just before Person1 speaks, the stack contains PLAN1 and PLAN2, as shown in Figure 11. (The top two plans have not yet been recognized, of course.)

The system is now in a state to recognize a continuation of PLAN1 (preference (1)), a clarification of PLAN2 or PLAN1 (preference (2)). Preference (3) would involve abandoning the current stack, so is very unlikely.

The passenger then asks "Where is it?", which is parsed into the action:

S-REQUEST (Person1, clerk1
INFORMREF (clerk1, Person1, ?term1,
EQUAL (?term1, loc (Gate7)))) (R2)

This analysis assumes the appropriate resolution of "it" to Gate7 by the focus mechanism of the communicative analysis. This makes the example simpler. The plan recognition would work even if the "it" were not resolved, and left as an unknown constant.

Although the clerk thought the INFORMREF of PLAN2 achieved the desired passenger KNOWREF, in actuality it did not provide a description enabling the passenger to execute the GOTO of PLAN1. Instead we have another request for clarification. The plan recognizer attempts to incorporate this utterance using the preferences described above. The first preference fails since the S-REQUEST does not match directly or by chaining any of the steps on the stack expected for execution. This preference, involving popping the completed PLAN2, is not possible because the utterance cannot be seen as a step of the BOARD plan. The second succeeds, and the utterance is recognized as part of an introduction of a new IDENTIFY-PARAMETER referring to the old one. This process is basically analogous to the process discussed in detail above, except that the plan to which the IDENTIFY-PARAMETER refers is found in the stack rather than constructed. The final results of the analysis are shown in Figure 11. Note the inclusion of S-INFORM, the clerk's actual realization of the INFORMREF I1.

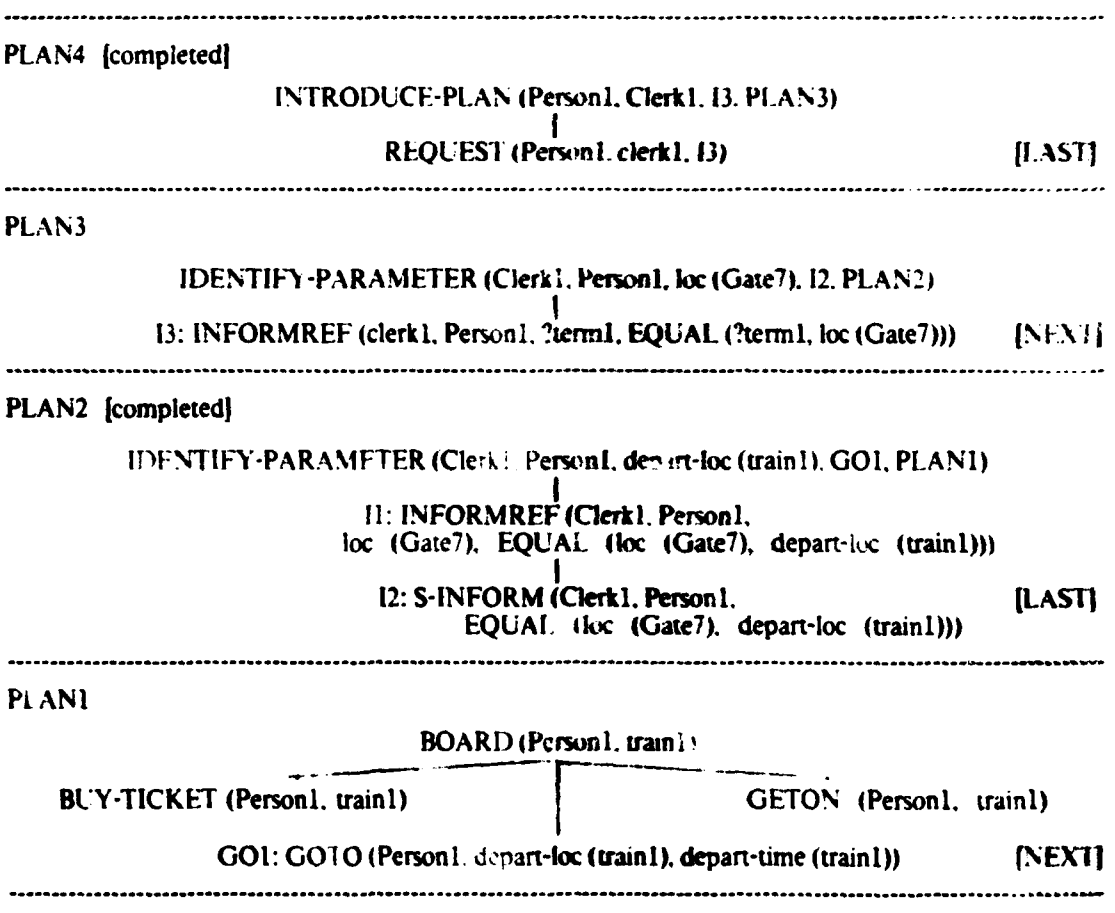


Figure 11: The Plan Stack after "Where is it?"

The clerk's reply is again hand simulated, using the INFORMREF in PLAN3, corresponding to "Down this way to the left--second one on the left." The system is ready to recognize the passenger's next plan which could include a meta-plan to the top IDENTIFY-PARAMETER (e.g., "Second what?") or a pop. The pop allows a meta-plan to the stacked IDENTIFY-PARAMETER of PLAN2 ("What's a gate?") or a pop, which allows a meta-plan to the original domain plan ("It's from Toronto?") or a pop. Since the original domain plan involved no communication, there are no utterances that can be a continuation of the domain plan itself. Each of these possible stacks is shown in Figure 12.

P: The eight-fifty to Montreal?
C: Eight-fifty to Montreal. Gate seven.
P: Where is it?
C: Down this way to the left. Second one on the left.

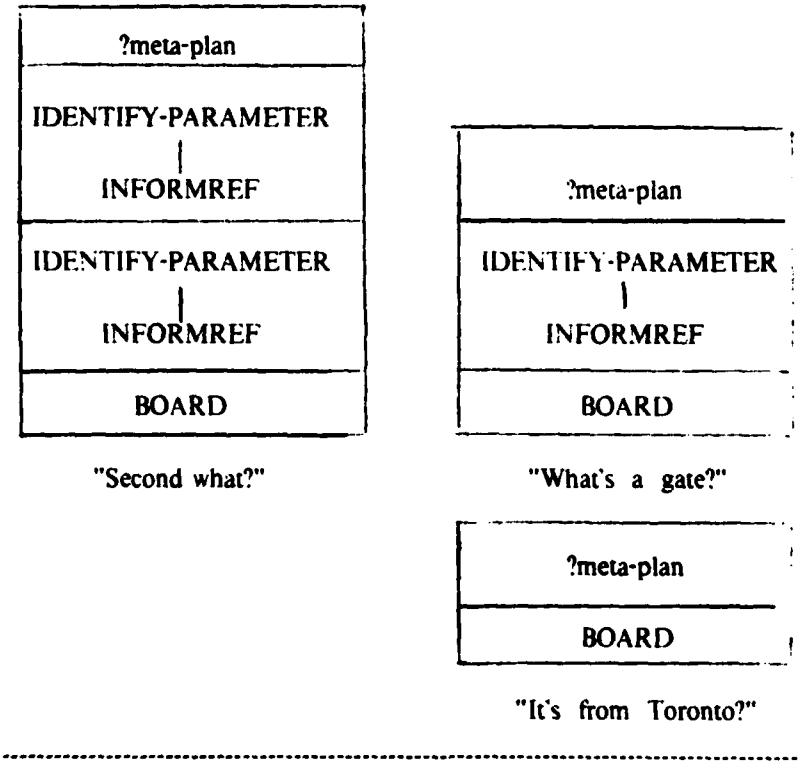


Figure 12: Coherent Dialogue Continuations of Preference Two

The dialogue actually concludes with the first preference, resumption of PLAN1 (as shown in Figure 11), reinforced by the passenger's "OK. Thank you." The "OK" is an example of a clue word [Reichman, 1981], words correlated with specific manipulations to the discourse structure. In particular, "OK" may indicate a pop [Grosz, 1977]. All but PLAN1 are then eliminated by "thank you," a discourse convention indicating termination of the dialogue. Note that unlike the previous utterance, what is going on with respect to the task plan can be quickly determined via communicative analysis.

5. Example: Dialogue 2

This section illustrates the system's processing of Dialogue 2 (repeated below), which introduces two new meta-plans and illustrates our treatment of indirect speech acts. This example shows the domain independence of our approach; although new domain plans must be introduced, the recognition procedure and the meta-plans remain unchanged. Further, since the model applies equally well to task-oriented and information-seeking domains, a bit of genre independence can also be claimed.

User: Show me the generic concept called "employee."
System: OK. <system displays network>
User: I can't fit a new ic below it. Can you move it up?
System: Yes. <system displays network>
User: OK, now make an individual employee concept whose ...
System: OK.

Dialogue 2

Recall that in Dialogue 2 the user interacts with a KL-ONE database system which is capable of graphically displaying KL-ONE concepts (KL-ONE [Brachman et al., 1979] is a knowledge representation language). Figure 13 presents the relevant domain plan schemas for this example. They are taken from [Sidner and Israel 1981, personal communication] with minor modifications and consist of plans to add new data into the network and to examine parts of the network. Both of these have a subplan involving the plan CONSIDER-ASPECT, in which the user considers some aspect of a network, for example by looking at it (the decomposition shown), listening to a description, or thinking about it. Again, our representation of action and time is greatly simplified for the purposes of this example.

HEADER: ADD-DATA (user, netpiece, data, screenLocation)
DECOMPOSITION: CONSIDER-ASPECT (user, netpiece)
PUT (user, data, screenLocation)

HEADER: EXAMINE (user, netpiece)
DECOMPOSITION: CONSIDER-ASPECT (user, netpiece)

HEADER: CONSIDER-ASPECT (user, netpiece)
DECOMPOSITION: DISPLAY (system, netpiece)

Figure 13: Domain Plans

Figure 14 presents CONTINUE-PLAN and CORRECT-PLAN, and repeats for convenience the meta-plans used in the example above. CONTINUE-PLAN takes an already existing plan (the WANT prerequisite) and moves execution to the next step. CORRECT-PLAN encodes requests which modify (debug) an already existing plan.

```
HEADER: INTRODUCE-PLAN (speaker, hearer, action, plan)
DECOMPOSITION: REQUEST (speaker, hearer, action)
EFFECTS: WANT (hearer, plan)
         NEXT (action, plan)
CONSTRAINTS: STEP (action, plan)
             AGENT (action, hearer)
```

```
HEADER: CONTINUE-PLAN (speaker, hearer, step, nextstep, plan)
PREREQUISITES: LAST (step, plan)
               WANT (hearer, plan)
DECOMPOSITION: achieve WANT (hearer, nextstep)
EFFECTS: NEXT (nextstep, plan)
CONSTRAINTS: STEP (step, plan)
             STEP (nextstep, plan)
             AFTER (step, nextstep)
             AGENT (nextstep, hearer)
```

```
HEADER: CORRECT-PLAN (speaker, hearer, laststep, newstep, nextstep, plan)
PREREQUISITES: WANT (hearer, plan)
               LAST (laststep, plan)
DECOMPOSITION-1: achieve WANT (hearer, newstep)
DECOMPOSITION-2: achieve WANT (hearer, nextstep)
EFFECTS: STEP (newstep, plan)
         AFTER (laststep, newstep)
         AFTER (newstep, nextstep)
         NEXT (newstep, plan)
CONSTRAINTS: STEP (laststep, plan)
             STEP (nextstep, plan)
             AFTER (laststep, nextstep)
             AGENT (newstep, hearer)
             ~CANDO (speaker, nextstep, plan)
             MODIFIES (newstep, laststep)
             ENABLES (newstep, nextstep)
```

```
HEADER: IDENTIFY-PARAMETER (speaker, hearer, parameter, action, plan)
DECOMPOSITION: INFORMREF (speaker, hearer, term, proposition)
EFFECTS: NEXT (action, plan)
         KNOW-PARAMETER (hearer, parameter, action, plan)
CONSTRAINTS: PARAMETER (parameter, action)
             PARAMETER (parameter, proposition)
             PARAMETER (term, proposition)
             STEP (action, plan)
             WANT (hearer, plan)
```

Figure 14: Meta-plan Schemas

More specifically, CORRECT-PLAN takes a pre-existing plan having subparts which do not interface as expected during execution; the plan thus needs to be modified by adding a new goal to restore the expected interactions. The pre-existing plan has subparts laststep and nextstep, where laststep was supposed to enable the performance of nextstep, but in reality did not. Thus the plan must be corrected by adding newstep to the original plan, which enables the performance of nextstep and thus of the rest of the plan. As in INTRODUCE-PLAN, the modification can be introduced by a REQUEST for an as yet to be performed step. The constraints

capture the plan situation described above and should be self-explanatory, with the following exceptions. MODIFIES (action2, action1) means that action2 is a variant of action1, for example the same action with different parameters or a new action with the same effects. ENABLES (action1, action2) means that the problematic preconditions of action2 are in the effects of action1.

Note again that for the purposes of this paper we are assuming that all meta-plans will be achieved via verbal communication with another agent. Also note that CONTINUE-PLAN and CORRECT-PLAN cannot be performed unless a plan has been introduced (just as one cannot resume a topic unless it has been suspended), and that these plans are domain-independent since they apply in the train domain above as well.

The processing begins with the parser's analysis of "Show me the generic concept called 'employee'," where E1 stands for the generic concept called "employee":

S-REQUEST (user, system, DISPLAY (system, user, E1))

Following the plan recognition algorithm, the system finds that the utterance is a REQUEST which introduces some plan containing the display action (preference (3)). Since INTRODUCE-PLAN is not a step in any other plan chaining stops; since it is a meta-plan, recognition from the DISPLAY now occurs. Since the display action could be a step of the CONSIDER-ASPECT plan, which itself could be a step of either the ADD-DATA or EXAMINE plans, the domain plan remains ambiguous. Note that heuristics can not eliminate either possibility, since at the beginning of the dialogue any domain plan is a reasonable expectation. Chaining halts, all plans are expanded, and the two stacks shown in Figure 15 are created. As desired, we have constructed a plan stack based on domain-specific expectations.

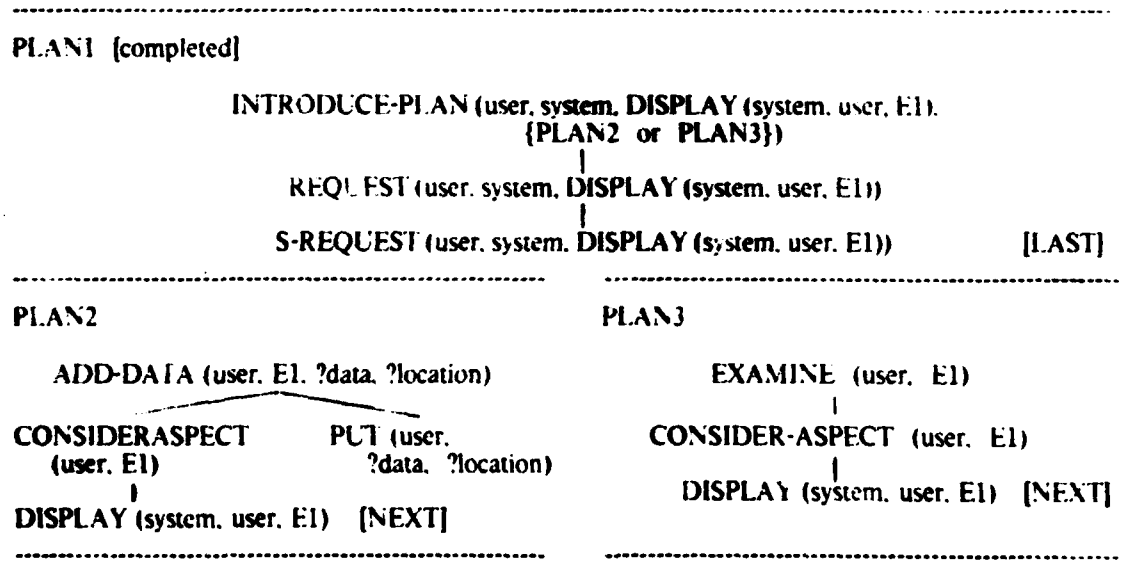


Figure 15: The Two Plan Stacks after "Show me the generic concept called 'employee'"

When the task structures are recognized, communicative analysis can note the global focus; since the discourse structure follows the task structure the executed S-REQUEST is marked as focused. Note that among the effects of any meta-plan is an updating of the focus in the plan referred to, in this case marking the DISPLAYs of plans 2 and 3 as focused. The system, assumed cooperative and a planner itself, examines the recognized plans and decides what to do (again, the planning our system performs is actually simulated). We simulate the system's choice to perform the display, generating "OK" to explicitly mark the step's completion. Since the REQUEST (and thus plan introduction) is completed, the stack can be popped and the execution of DISPLAY performed as a coherent next move. Also note that even though the recognized intent is ambiguous, the system can still act; deciding exactly what to do in such cases is an interesting planning issue.

The user's response, "I can't fit a new IC below it," is processed by the parser as S-INFORM (user, system, ~CANDO (user, FIT (user, ?ic, belowE1)). In other words, the fact that the system decided to perform the DISPLAY without knowing whether a PUT would follow (and if so what would be PUT) has now caused problems in the user's original plan, viz., the location of the node for the generic concept did not leave enough room to fit a new IC node below it.

The utterance is recognized by the plan recognizer as either an indirect REQUEST or a literal INFORM, and since neither is yet connected to any domain or meta-plan, we pursue both alternatives. From the indirect REQUEST we may postulate that the current plan is being modified, e.g., a meta-plan to one of the candidate current plans is introduced (preference (2)), as shown in Figure 16. Recognition goes from the S-INFORM to the REQUEST to the candidate plan CORRECT-PLAN (user, system, DI, ?action, FIT1, PLAN2), where FIT is an instance of PUT (we will use the terms interchangeably). The parameter ?action in CORRECT-PLAN is bound to various system actions which modify the display to enable the fit, and is inserted into PLAN2 (and marked as focused). DISPLAY and PLAN2 are known via precondition and constraint satisfaction. Note that the REQUEST corresponding to introduction of a new plan is discarded since it does not tie in with any of the expectations (and a preference (2) choice is preferred over a preference (3) choice). Any other possibilities (a modification of PLAN3, a continuation or IDENTIFY-PARAMETER of PLAN3 or PLAN2, or a correction of PLAN2 with the fit as the new (instead of the next) step all fail due to constraint violation. Thus only one of the stacks shown in Figure 15 remains plausible.

PLAN4

CORRECT-PLAN (user, system, D1, ?action, FIT1, PLAN2)
|
REQUEST (user, system, FIT1)
|
S-INFORM (user, system, ~CANDO (user, FIT1)) [LAST]

where AGENT (?action, system)
MODIFIES (?action, D1)
ENABLES (?action, FIT1)

PLAN2

ADD-DATA (user, E1, ?ic, belowE1)

CONSIDERASPECT (user, E1) ?action [NEXT] FIT1: FIT (user, ?ic, belowE1)
|
D1: DISPLAY (system, user, E1) [LAST]

Figure 16: The Plan Stack after "I can't fit a new IC below it"

The parser's output from the user's continuation, "Can you move it up?" is

S-REQUEST (user, system INFORMIF (system, user
(CANDO (system, MOVE (system, E1, up))))),

recognized as either a literal or indirect REQUEST. From the indirect REQUEST we can chain via its KNOW effect to INFORMREF and then to IDENTIFY-PARAMETER, as shown in Figure 17. PLAN4 can now be considered completed since it is fully instantiated and its steps all performed. Even without our preference ordering, the other possibilities are eliminated, the IDENTIFY-PARAMETER act, obtained by chaining from the literal REQUEST, cannot tie into PLAN4; and corrections to and continuation of PLAN4 can't be well-formed since PLAN4 does not have multiple actions. Finally, the introduction of a new plan would only be considered as a last resort since it would not tie into the existing stack.

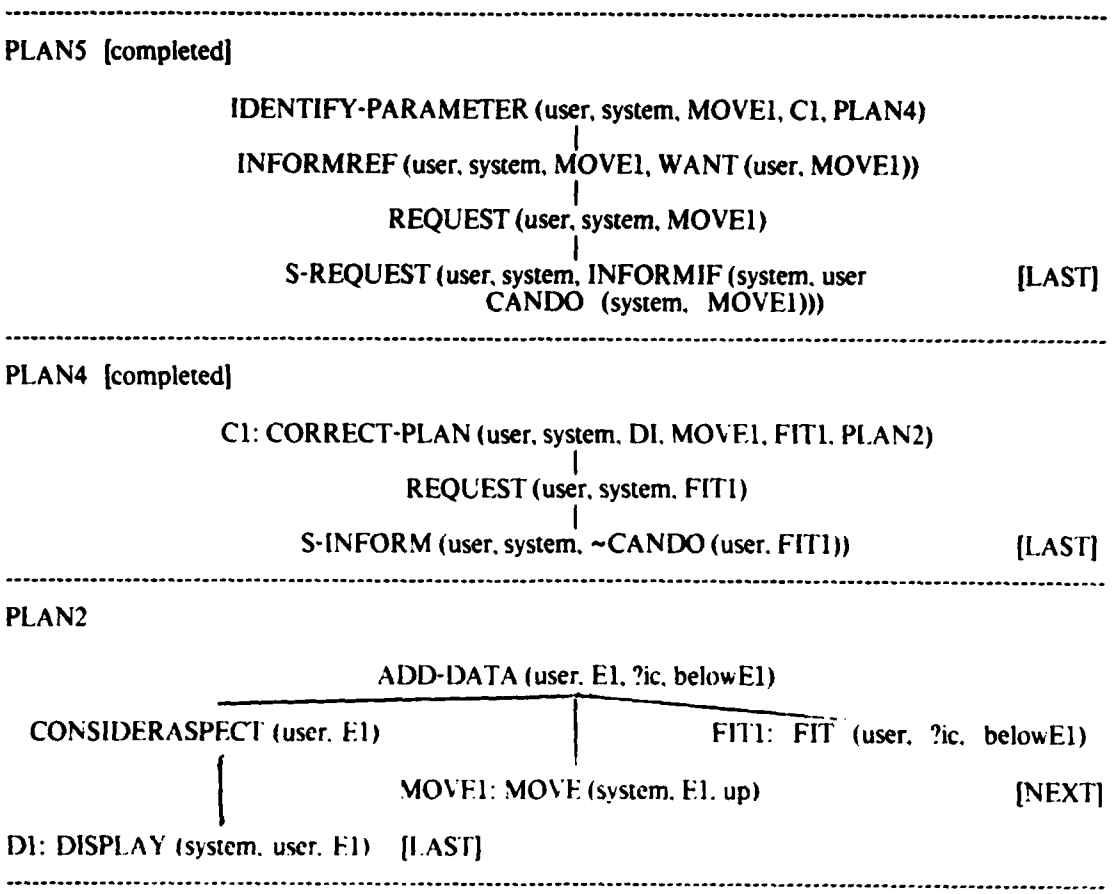


Figure 17: The Plan Stack after "Can you move it up?"

The system is simulated as follows: noting that the user's IDENTIFY-PARAMETER and CORRECT-PLAN are both complete, it pops the stack and resumes PLAN2 with the new step inserted by the meta-plan. Once MOVE1 is done, the system is now ready to recognize the continuation of the plan (i.e., FIT1 by the user).

Before the user's next utterance ("OK, now make ...") is even processed, the system's discourse mechanism picks up the two initial clue words which explicitly marks this continuation with the next step. The rest of the utterance is then recognized (analagously to the above detailed explanation) as a continuation meta-plan, as shown in Figure 18. At this stage, it would be appropriate for the system to pop the completed top plan and resume PLAN2 by performing MAKE1.

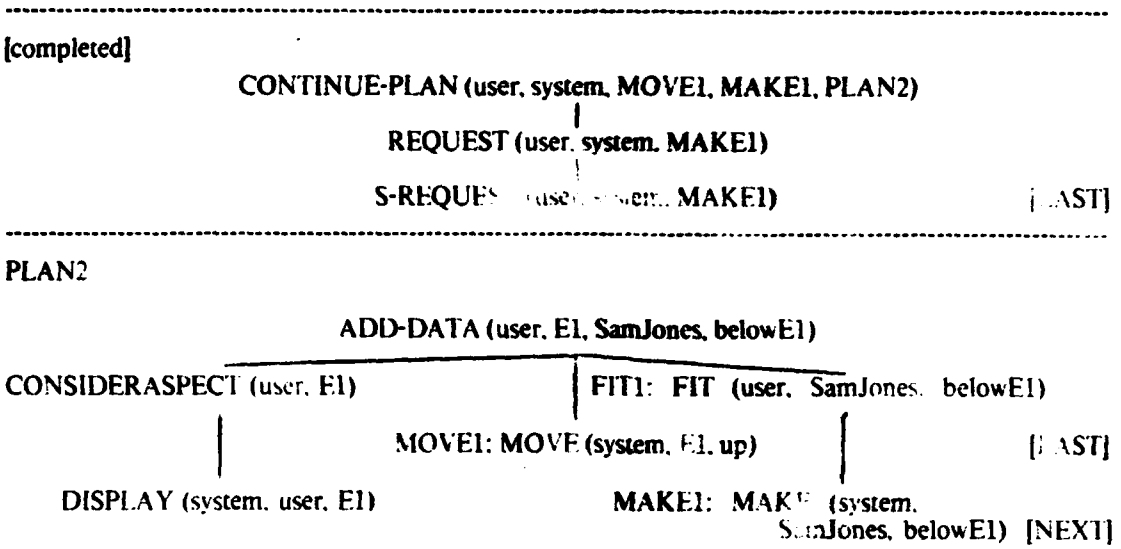


Figure 18: "OK, now make an individual concept whose ..."

Note how the framework would still prove useful if the user's "Can you move it up?" was preceded by the system's "What do you want me to do?". The analysis is shown in Figure 19. Comparing this with the situation in Figure 17, note that after PLAN6 is introduced the analysis will be the same as for the non-prompted case.

[completed]

INTRODUCE-PLAN (system, user, I4, PLAN6)
|
REQUEST (system, user, I4)
|
S-REQUEST (system, user, I4) [LAST]

PLAN6

IDENTIFY-PARAMETER (user, system, ?action, C1, PLAN4)
|
I4: INFORMREF (user, system, ?action, WANT (user, ?action))) [NEXT]

with AGENT (system, ?action)

PLAN4

C1: CORRECT-PLAN (user, system, D1, ?action, FIT1, PLAN2)
|
REQUEST (user, system, FIT1)
|
S-INFORM (user, system, ~CANDO (user, FIT1)) [LAST]

PLAN2

ADD-DATA (user, E1, ?ic, belowE1)

CONSIDERASPECT (user, E1) ?action [NEXT] FIT1: FIT (user, ?ic, belowE1)
|
D1: DISPLAY (system, user, E1) [LAST]

Figure 19: If the system had said "What do you want me to do?"

It is also interesting to note what kind of analysis would result if the user's two utterances were reversed. After the user says "Can you move it up?" CORRECT-PLAN (user, system, D1, MOVE1, FIT (user, ?data, ?location), PLAN2) is recognized and PLAN7 is put above PLAN2 on the stack. Notice that the system knows both what to do and why, since from the constraint the system can infer that the interaction between DISPLAY and FIT is the problem. This was not true when CORRECT-PLAN was first recognized in the original dialogue (Figure 16). The user explicitly confirms this with the next utterance, "I can't fit a new ic below it," recognized as an IDENTIFICATION of the FIT parameter in CORRECT-PLAN which fills in the unknown parameters of the FIT. The state of the stack after these two utterances would be as shown in Figure 20. Comparison of Figures 17 and 20 illustrate how the same utterances are analyzed differently depending on the discourse context.

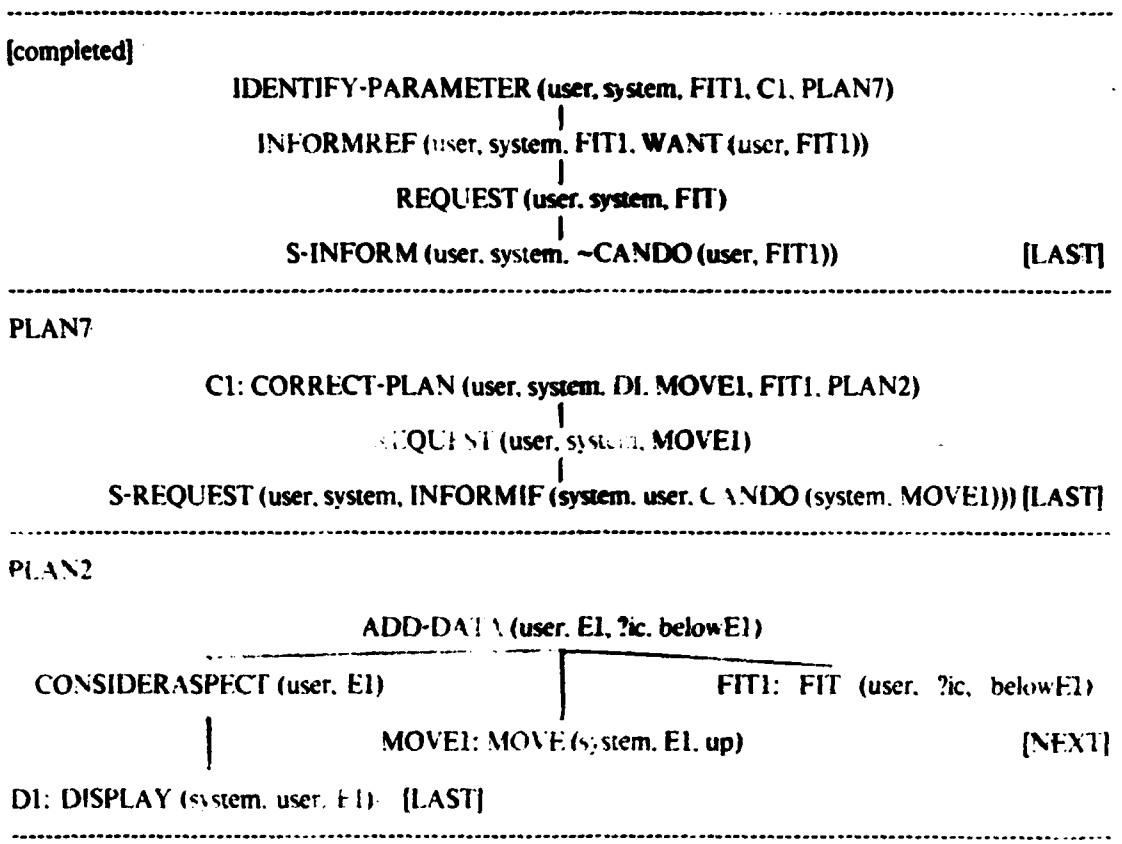


Figure 20: If the User's Utterances were Reversed

6. Comparisons with Other Work

6.1 Recognizing Speech Acts

The major difference between our present approach and previous plan recognition approaches to speech acts (e.g., [Allen and Perrault, 1980]) is that we have a hierarchy of plans, whereas all the actions in Allen and Perrault were contained in a single plan. This has enabled us to simplify the notion of what a plan is and to solve a puzzle that arose in the one-plan systems. Consider how the plan recognizer of Allen and Perrault [1980] connects an observed speech act to an expected (BOARD) domain goal, as in Figure 21.

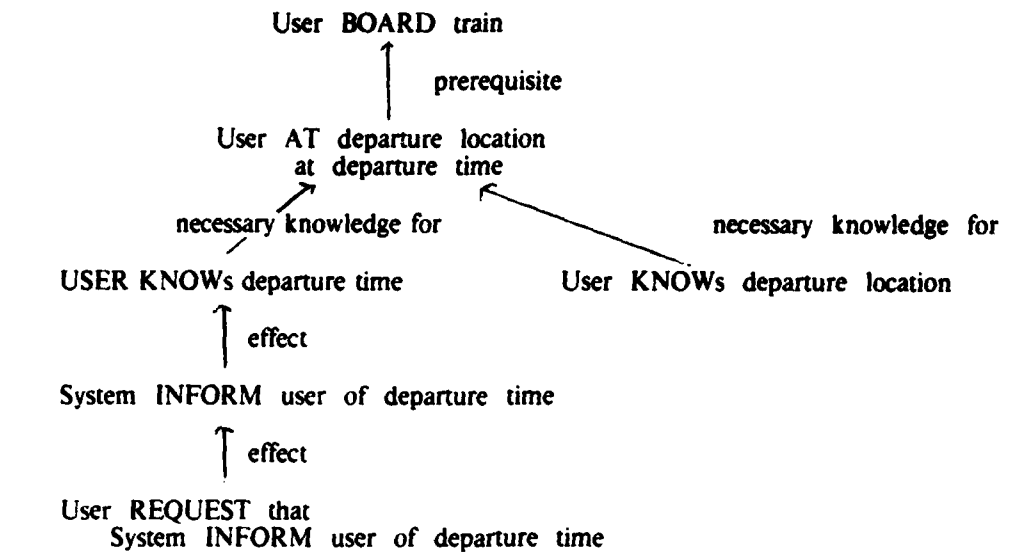


Figure 21: Simple Plan Recognized from "When does the Montreal train leave?" [Allen & Perrault, 1980]

In such systems, plans were networks of action and state descriptions linked by causality and subpart relationships, plus a set of knowledge-based relationships. This latter set was not categorized as either a causal or a subpart relationship, and so needed a special mechanism. Incorporating this into the action definitions would have required having a knowledge precondition for every term in every action. The problem was that these relationships were not part of any plan itself, but a relationship between plans.

In our system, this relationship between plans is explicit, eliminating the need for the plan recognizer's special mechanism. The "knowref" and "know-pos" and "know-neg" relations are modeled as constraints between a plan and a meta-plan, i.e., the plan to perform the task and the plan to obtain the knowledge necessary to perform the task.

Besides simplifying what counts as a plan, the multiplan approach provides some insight into how much of the user's intentions must be recognized in order to respond appropriately. We suggest that the top plan on the stack must be connected to a discourse goal (e.g., a meta-plan). The lower plans may be only partially specified, and be filled in by later utterances. An example of this appears in considering Dialogue 2 above.

Regarding the analysis of indirect speech acts (*ISA*), in our present system we have a set of decompositions that correspond to the conventional ISA. These are abstractions of inference paths that can be derived from first principles as in Allen and Perrault. Similar "compilation" of ISA can be found in Sidner and Israel [1981] and Carberry [1983]. It is not clear in those systems, however, whether the literal interpretation of such utterances could ever be recognized. In their systems, the ISA

analysis is performed before the plan recognition phase. In our system, the presence of "compiled" ISA allows indirect forms to be considered easily, but they are just one more option to the plan recognizer. The literal interpretation is still available and will be recognized in appropriate contexts. As far as the task is concerned, whether a request was indirect or direct is irrelevant. For example, if we set up a plan to ask about someone's knowledge (say, by an initial utterance of "I need to know where the schedule is incomplete"), then the utterance "Do you know when the Windsor train leaves?" is interpreted literally as a yes/no question because that is the interpretation explicitly expected from the analysis of the initial utterance.

Sidner and Israel [1981] outlined an approach that extended Allen and Perrault in the direction we have done as well. They allowed for multiple plans to be recognized but did not appear to relate the plans in any systematic way. Section 5 presented our analysis of (a minor variant of) one of their dialogues. The major difference is our use of meta-plans and discourse knowledge. Although Sidner and Israel [1981] did not exploit surface linguistic phenomena, their model is being extended in that direction [Sidner et al., 1984].

A comparison with their analysis of "I can't fit a new ic below it" illustrates the advantages of the meta-plan system. Upon receiving the parser's analysis of the utterance, their system uses a default rule which maps "I can't x" to "I want x." Basically, this just retrieves the conventionalized meaning of the indirect speech act as described above, which is then recognized as part of a plan structure in the normal way. However, there is another rule which is also triggered by the original utterance: if a user can't do something he or she wants, the user will execute an "unblock" (i.e., debug) plan, temporarily suspending the blocked plan. Unfortunately, the details of how these two plans are related and how the ADD-DATA suspension is managed are left unexplored. Our meta-plan mechanism eliminates the need for special default rules outside of the plan recognition process and provides a mechanism for managing topic suspensions and resumptions in the general case.

Grosz [1979], Levy [1979], and Appelt [1981] extended the planning framework to incorporate multiple perspectives, including both communicative and task goal analysis. However, they did not present details for dialogues. ARGOT [Allen et al., 1982] was an attempt to extend the work to dialogue processing, and led to the development of what has been presented here.

Pollack [1984] is extending plan recognition for understanding in the domain of dialogues with experts; she abandons the assumption that people always know what they really need to know in order to achieve their goals. In our work we have implicitly assumed appropriate queries and have not yet addressed this issue.

Wilensky's use of meta-planning knowledge [1983] enables his planner to deal with goal interaction. For example, he has meta-goals such as resolving goal conflicts and eliminating circular goals. This treatment is similar to ours except for a matter of emphasis. His meta-knowledge is concerned with his planning mechanism, whereas our meta-plans are concerned with acquiring knowledge about plans, interacting with other agents, and shifting focus. The two approaches are also similar in that they use the same planning and recognition processes for both plans and meta-plans.

6.2 Discourse

Although both Sidner and Israel [1981] and Carberry [1983] have extended the Allen and Perrault paradigm to deal with task plan recognition in extended dialogues, neither system currently performs any explicit discourse analysis. As described earlier, Carberry does have a (non-discourse) tracking mechanism similar to that used in [Grosz, 1977]; however, the mechanism cannot handle topic switches and resumptions, nor use surface linguistic phenomena to decrease the search space. Since she is concerned with tracking goals in an information-seeking domain, one in which a user seeks information in order to formulate a plan which will not be executed during the dialogue (this is similar to what happens in our train domain), her recognition procedure is also not as tied to the task structure. Supplementing our model with meta-plans provided a unifying (and cleaner) framework for understanding in both task-oriented and information-seeking domains.

Reichman [1981] and Grosz [1977] used a dialogue's discourse structure and surface phenomena to mutually account for and track one another. Grosz concentrated on task-oriented dialogues with subdialogues corresponding only to subtasks (e.g., no clarifications). Reichman was concerned with a model underlying all discourse genres. However, although she distinguished communicative goals from speaker intent her research was not concerned with either speaker intent or any interactions between goals and intents. Since our system incorporates both types of analysis, we have not found it necessary to perform complex communicative goal recognition as advocated by Reichman. Knowledge of plans and meta-plans, linguistic surface phenomena, and simple discourse conventions have so far sufficed. This approach appears to be more tractable than the use of rhetorical predicates and coherence relations (e.g., ELABORATE, SUPPORT, ...) advocated by Reichman and others such as Mann [1984], Hobbs [1979], and McKeown [1982].

Carbonell [1982] suggests that any comprehensive theory of discourse must address issues of meta-language communication, as well as integrate the results with other discourse and domain knowledge, but does not outline a specific framework. We have developed a computational model which addresses many of these issues for an important class of dialogues.

7. Summary

We have presented a plan-based natural language model incorporating both task and communicative analysis, summarized in Figure 22. This paper has emphasized the task analysis, in particular our meta-plans, the plan stacks, and a plan recognizer able to use these structures. We have shown how such a model can be used to understand clarification and debugging dialogues, e.g., dialogues which arise from plan execution difficulties and which exhibit the recursiveness characteristic of spontaneous conversation, using examples from two domains and conversational genres. Due to space limitations, our approach to discourse, as well as the task/communicative interactions, have only been touched on.

TASK ANALYSIS	COMMUNICATIVE ANALYSIS
Plan Structures:	
domain plans	focus mechanism
meta-plans	cue words
plan stacks	mode of reference
Plan Recognition	

Figure 22: Summary

Obviously, many issues remain to be addressed. Our dialogues are still fairly rigid (task-oriented): it will be interesting to see how our model handles topic suspension and resumption in the general case. Except for our handling of the two-sentence utterance in Dialogue 2, we have largely ignored the narrative (as opposed to conversational) aspects of our work, a topic of much interest in the language generation community. Of course, all the issues of language generation in this model remain to be addressed. Finally, we must remove our mutual belief assumption to understand dialogues in which the conversants' dialogue and/or world models differ.

Acknowledgements

We would like to thank Henry Kautz, Tom Blenko, Pat Hayes, Ron Loui, Brad Miller, David Sher, Dan Russell, Marc Vilain, and Jay Weber for their comments on versions of this paper. Mark Guiliano for his work on Horne, and Jay Weber for his work on the parser.

8. References

Allen, J.F., A.M. Frisch, and D.J. Litman, "ARGOT: The Rochester dialogue system," *Proc.*, Nat'l. Conf. on Artificial Intelligence, Pittsburgh, August 1982.

Allen, J.F., M. Giuliano, and A.M. Frisch, "The HORNE Reasoning System," TR 126, Computer Science Dept., U. Rochester, 1983.

Allen, J.F. and C.R. Perrault, "Analyzing intention in utterances," TR 50, Computer Science Dept., U. Rochester, 1979; *Artificial Intelligence 15*, 3, Dec. 1980.

Appelt, D.E., "Planning natural language utterances to satisfy multiple goals," Ph.D. thesis, Stanford U., 1981.

Brachman, R.J., R.J. Bobrow, P.R. Cohen, J.W. Klovstad, B.V. Webber, and W.A. Woods, "Research in natural language understanding: Annual report," BBN Report 4274, Cambridge, MA, 1979.

Brown, J. and R. Burton, "Semantic grammar: A technique for constructing natural language interfaces to instructional systems," BBN Report 3587, May 1977.

Carberry, S., "Tracking user goals in an information seeking environment," *Proc.*, Nat'l. Conf. on Artificial Intelligence, 1983.

Carbonell, J.G., "Meta-language utterances in purposive discourse," TR 125, Computer Science Dept., Carnegie-Mellon U., June 1982.

Cohen, P.R. and C.R. Perrault, "Elements of a plan-based theory of speech acts," *Cognitive Science 3*, 3, 1979.

Cohen, R., "A computational model for the analysis of arguments," Ph.D. thesis and TR 151, Computer Science Dept., U. Toronto, October 1983.

Fikes, R.E. and N.J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence 2*, 189-205, 1971.

Grosz, B.J., "The representation and use of focus in dialogue understanding," TN 151, SRI, July 1977.

Grosz, B.J., "Utterance and objective: Issues in natural language communication," *Proc.*, IJCAI, 1979.

Hobbs, J.R., "Coherence and coreference," *Cognitive Science 3*, 1, 1979.

Horrigan, M.K., "Modelling simple dialogs," TR 108, U. Toronto, May 1977.

Levy, D., "Communicative goals and strategies: Between discourse and syntax," in T. Givon (ed). *Syntax and Semantics* (vol. 12). New York: Academic Press, 1979.

Litman, D.J., "Discourse and problem solving," Report 5338, Bolt Beranek and Newman, July 1983; TR 130, Computer Science Dept., U. Rochester, Sept. 1983.

Mann, W.C., "Discourse structure for text generation," *Proc.*, 10th Int'l. Conference on Computational Linguistics, Stanford, July 1984.

McKeown, K.R., "Generating natural language text in response to questions about database structure," Ph.D. thesis, U. Pennsylvania, 1982.

Pollack, M.E., "Goal inference in expert systems," Ph.D. thesis proposal, U. Penn., January 1984.

Polanyi, L. and R.J.H. Scha, "The syntax of discourse," *Text 3*, 3, 1983.

Reichman, R., "Plain speaking: A theory and grammar of spontaneous discourse," Report 4681, Bolt, Beranek and Newman, Inc., 1981.

Sacerdoti, E.D. *A Structure for Plans and Behavior.* New York: Elsevier, 1977.

Sidner, C.L., "Focusing in the comprehension of definite anaphora," in M. Brady (ed). *Computational Models of Discourse.* Cambridge, MA: MIT Press, 1983.

Sidner, C.L. and M. Bates, "Requirements for natural language understanding in a system with graphic displays," Report 5242, Bolt Beranek and Newman, Inc., 1983.

Sidner, C.L., B. Goodman, A. Haas, M. Moser, D. Stollard, and M. Vilain, "Research in knowledge representation for natural language understanding," BBN Report 5694, 1984.

Sidner, C.L. and D. Israel, "Recognizing intended meaning and speakers' plans," *Proc.*, 7th IJCAI, Vancouver, B.C., August 1981.

Wilensky, R. *Planning and Understanding.* Addison-Wesley, 1983.

END

FILMED

3-85

DTIC